計測自動制御学会東北支部 第 301 回研究集会 (2016.5.23)
資料番号 301-10

MATLAB によるスパースフーリエ変換の実装と性能評価

A MATLAB Implementation of the Sparse Fourier Transform and Its Performance Evaluation

○田中 健也，阿部 正英，川又 政征

○ Kenya Tanaka, Masahide Abe, Masayuki Kawamata

東北大学

Tohoku University

キーワード： スパースフーリエ変換 (Sparse Fourier Transform)，スパース信号 (Sparse Signal)，高速フーリエ変換 (Fast Fourier Transform)，近似アルゴリズム (Approximation Algorithm)

連絡先： 連絡先　〒 980-8579　仙台市青葉区荒巻字青葉 6-6-05
東北大学　大学院工学研究科　電子工学専攻　川又・阿部（正）研究室
田中 健也，Tel.: (022)795-7095，Fax.: (022)263-9169，E-mail: kenya@mk.ecei.tohoku.ac.jp

1. まえがき

近年，センサー技術の向上やビックデータ処理需要の高まりに伴い，より長い信号をより高速に処理する必要性が高まっている．非常に長い信号に対しては，高速フーリエ変換でも十分に高速とは言えない場合がある．本論文では，高速フーリエ変換を用いても十分に高速とは言えない非常に長い信号に対し，より高速に実行可能な「スパースフーリエ変換」を MATLAB で実装し，その性能の評価を行う．

第 2 章では，本論文で用いる数学的な表記および用語の定義を記す．第 3 章では，離散フーリエ変換の定義および高速フーリエ変換の概要を説明する．第 4 章では，スパースフーリエ変換のアルゴリズムを説明する．第 5 章では，スパースフーリエ変換の MATLAB への実装について記す．第 6 章では，MATLAB で実装したスパースフーリエ変換を用いて実行時間と推定精度に関する評価を行う．第 7 章では，本論文全体のまとめを記す．

2. 表記および用語の定義

本論文で用いる数学的な表記および用語について，以下にその定義を記す．

$\mathrm{Re}\,[z]$： 複素数 z の実数部を表す．

$\mathrm{Im}\,[z]$： 複素数 z の虚数部を表す．

$n \bmod m$： n を m で割ったときの剰余を表す．

$\mathrm{round}(n)$： n に最も近い整数を表す．

$\mathcal{O}(n)$： オーダーが n であることを表す．

3. 離散フーリエ変換

本章では離散フーリエ変換 (DFT) の概要について述べる [1)2)]．信号 $\boldsymbol{x} \in \mathbb{C}^N$ の離散フーリエ変換 $\boldsymbol{X} \in \mathbb{C}^N$ は，

$$X(m) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi mn}{N}} \qquad (1)$$

で定義される *．振幅スペクトル $|X(m)|$ は，$\boldsymbol{x}$ を構成する複素指数関数の周波数ごとの振幅を表す．位相スペクトル $\angle X(m)$ は，$\boldsymbol{x}$ を構成する複素指数関数の周波数ごとの位相を表す．また，離散フーリエ逆変換 (IDFT) は，

$$x(n) = \sum_{m=0}^{N-1} X(m)e^{j\frac{2\pi mn}{N}} \qquad (2)$$

で定義される *．

N 点離散フーリエ変換の直接計算の計算量は $\mathcal{O}(N^2)$ である．直接計算では N が大きい場合に計算量が膨大になってしまうため，高速に実行することが困難となる．そこで，一般的には計算量が $\mathcal{O}(N\log N)$ である高速フーリエ変換 (FFT) という手法が用いられるが，高速フーリエ変換を用いても信号長 N の増加が計算量に大きく影響するため，非常に長い信号に対しては十分に高速とは言えない．したがって，信号長 N が非常に大きい場合にも高速に計算可能なアルゴリズムが求められる．

4. スパースフーリエ変換

信号に含まれる主要なスペクトルがまばらである信号をスパース信号と呼ぶ [3)]．Fig.1 にスパース信号の例を示す．スパース信号に含まれる主要なスペクトルの本数を sparsity と呼び，k で表す．

スパースフーリエ変換 (SFFT) は，スパース信号に対して高速にフーリエ変換を行う手法である．スパース信号の主要なスペクトルのみに着目して計算するため，非常に長い信号に対しても高速に処理を行うことが可能である．また，本論文ではスパースフーリエ変換で扱う信号長 N を 2 のべき乗であると仮定する．

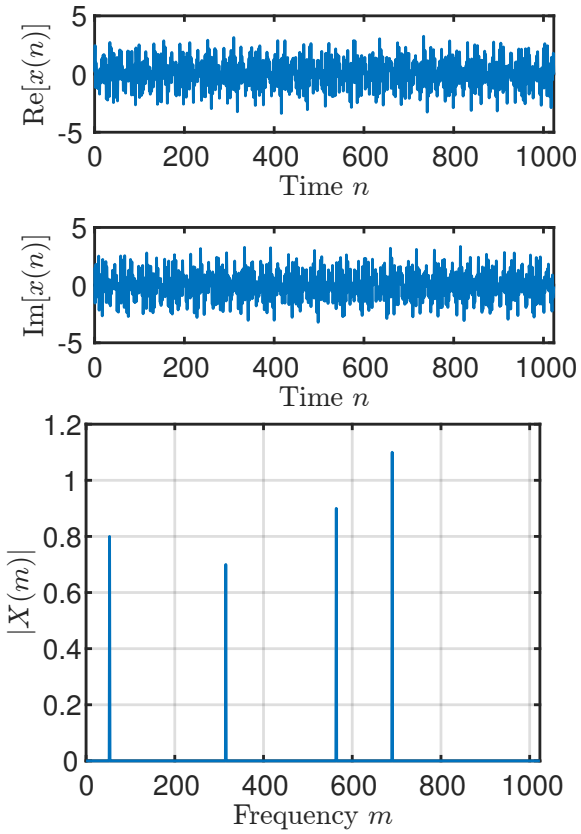

Fig. 1　複素スパース信号 $(N = 1024, k = 4)$

4.1 アルゴリズムの概要 [3)]

スパースフーリエ変換では，まず入力信号に対して Fig.2 に示すような窓関数 $\boldsymbol{w}$ によって時間領域で窓かけを行う．この窓関数は，Dolph-Chebyshev 窓と矩形な周波数特性を持つ窓を掛けることで得られる．一般的に，スパースフーリエ変換に用いる窓関数は時間領域でのサポート（値を持つインデックス幅）が小さく，急峻な減衰特性を持つものが良いとされる．

ここから，スパースフーリエ変換の入力として Fig.1 に示す複素スパース信号 $\boldsymbol{x} \in \mathbb{C}^N$ を用いて説明をする．この信号に対して窓かけをす

*本論文では，離散フーリエ変換の際の $\frac{1}{N}$ 倍による正規化を順変換の際に行う．この場合，逆変換の際に正規化を行う MATLAB とは直接対応しないため，注意が必要である．

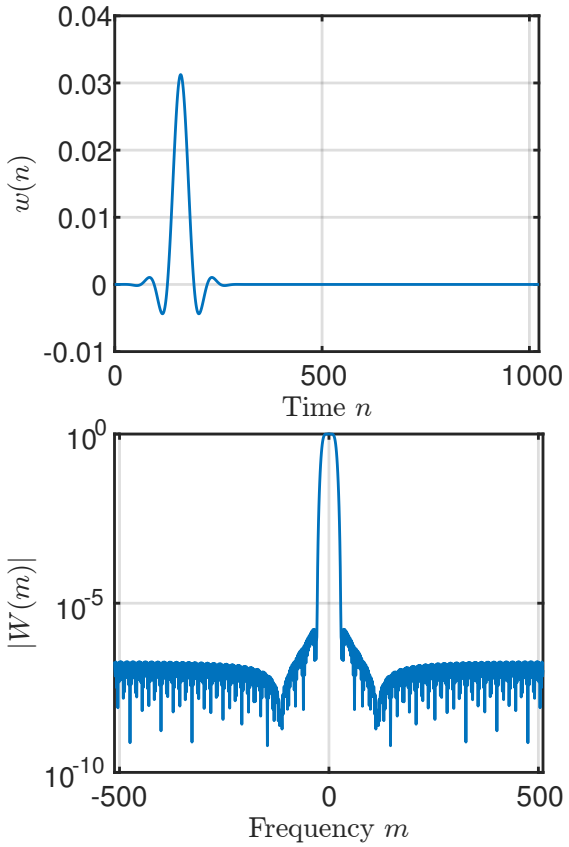

Fig. 2　窓関数

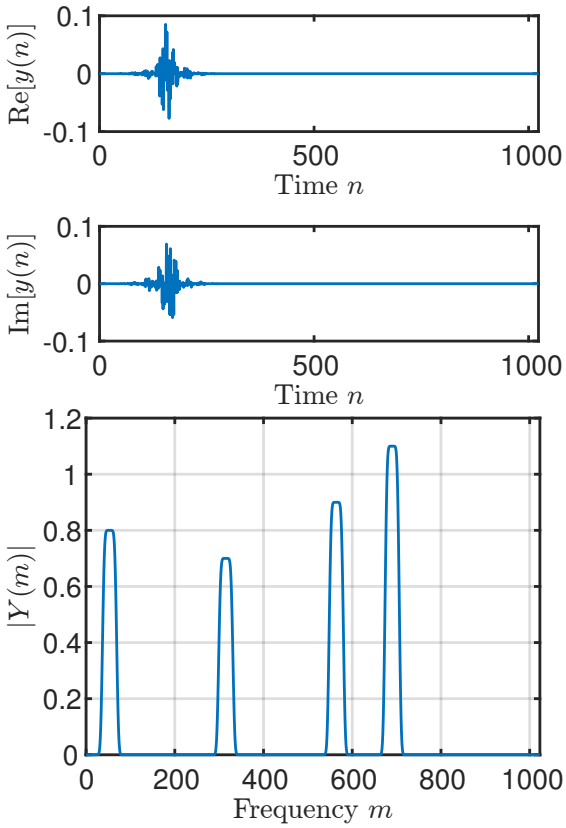

Fig. 3　窓かけ後の信号

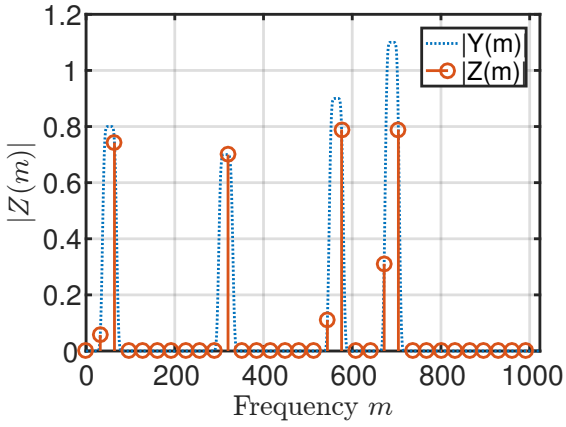

Fig. 4　Subsampled-DFT

ると，Fig.3 に示す信号 $\boldsymbol{y}$ が得らる．この信号は周波数領域ではスペクトルが広がった形になっている．

次に $\boldsymbol{y}$ に対し，Subsampled-DFT(Subsampled-FFT) によって周波数スペクトル $\boldsymbol{Z} \in \mathbb{C}^B$ を得る．ここで，Subsampled-DFT は，

$$Z(m) = \sum_{n=0}^{B-1} \sum_{p=0}^{\frac{N}{B}-1} y(n+Bp) \mathrm{e}^{-j\frac{2\pi mn}{B}} \quad (3)$$

で定義される．ここで，B は，

$$B = 2^{\mathrm{round}(\log k)+3} \quad (4)$$

で定義し，これは窓関数の設計や Subsampled-DFT と密接に関わる重要な定数である．また，$\boldsymbol{Z}$ は以下の関係を満たす（Fig.4）．

$$Z(m) = Y\left(\frac{N}{B}m\right) \quad (5)$$

ここから，Subsampled-DFT によって得られた周波数スペクトル $\boldsymbol{Z}$ を用いて原信号の主要な周波数スペクトルの位置（インデックス）を推定する．Fig.1 と Fig.4 から，原信号の主要なスペクトルは，$\boldsymbol{Z}$ の主要なスペクトルの周辺にあると推定できる（Fig.5）．しかし，この推定では大まかな範囲しか分からず，原信号の主要なスペクトルの正確な位置を求めることはできない．次の 4.2 節では，本節で紹介した手法を発展させ，より正確な推定を行うための手法を紹

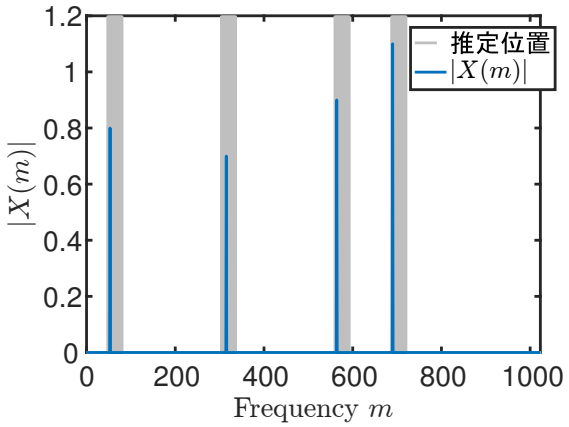

Fig. 5　$\boldsymbol{Z}$ による位置の推定結果

介する．

4.2　正確な推定のためのアルゴリズム [3)]

本節では前の 4.1 節で説明したアルゴリズムに改良を加え，より正確な推定を行うための手法を紹介する．

位置の推定結果を変化させるために，信号に窓かけを行う前に時間領域での並び変えを行う．信号 $\boldsymbol{x} \in \mathbb{C}^N$ の並び替え $\boldsymbol{x}_\mathrm{p} \in \mathbb{C}^N$ を，

$$x_\mathrm{p}(n) \quad = \quad x(rn \bmod N) \qquad (6)$$

と定義する．ここで，r は並び替えに影響するパラメータで，$[1, N-1]$ からランダムで選んだ奇数である *．スケーリング法則 [3)] より，$\boldsymbol{x}_\mathbf{p}$ の周波数スペクトルは，

$$X_\mathrm{p}(m) \quad = \quad X(rm \bmod N) \qquad (7)$$

となる．すなわち，ここでは時間領域で信号を並び替えることで，周波数スペクトルの位置を並び替えるという作業を行っている（Fig.6）．

並び替え後の信号 $\boldsymbol{x}_\mathrm{p}$ に対し，4.1 節の手法（窓かけ，Subsampled-DFT）によって周波数スペクトルを求め，原信号の主要なスペクトルの位置を推定する．推定結果を Fig.7 に示す．

*並び替えパラメータ r は式 (7) より，r の値と信号長 N は互いに素ではない場合，並び替えたスペクトルが同じインデックスで重なってしまう可能性がある．しかし，今回は信号長 N を 2 のべき乗と仮定しているため，r は単純に奇数の値を選べばよい．

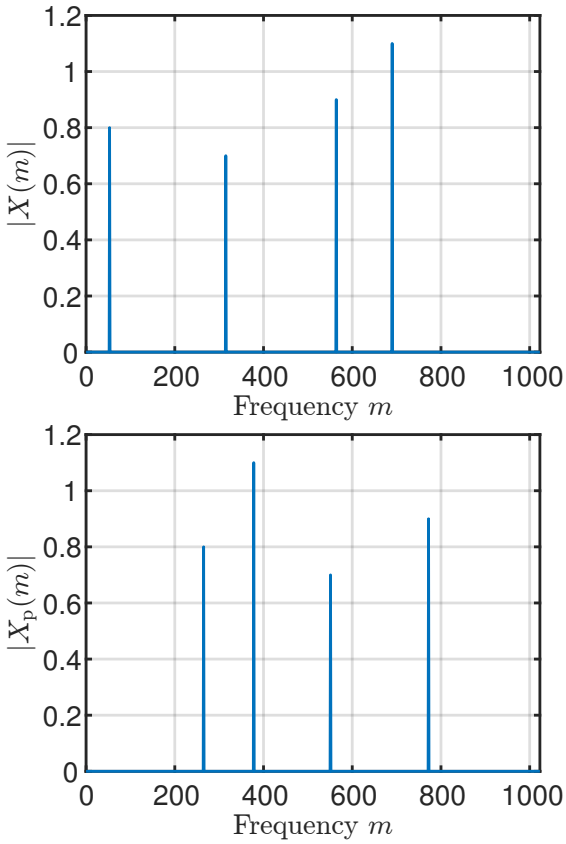

Fig. 6　並び替えによるスペクトルの変化

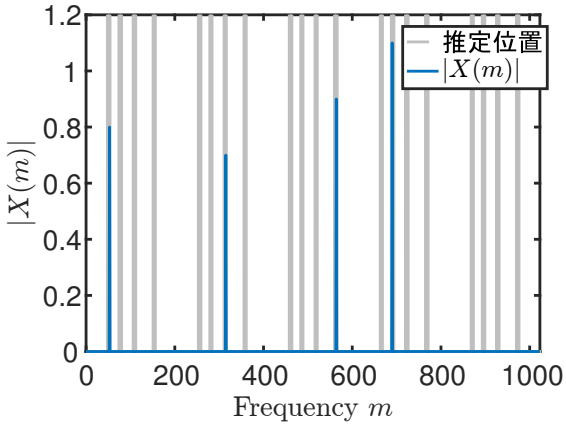

Fig. 7　並び替えを行った場合の推定結果

Fig.5 と Fig.7 を比較すると，信号の並び替えの影響で位置推定の結果が異なっているのが分かる．この二つの位置推定結果から，より多く推定された位置のみを取り出すことで Fig.8 の結果が得られる．すなわち，二つの結果を総合することで，不要な位置を排除し，推定結果の精度を上げることが可能である．

したがって，より高い精度で原信号の主要な

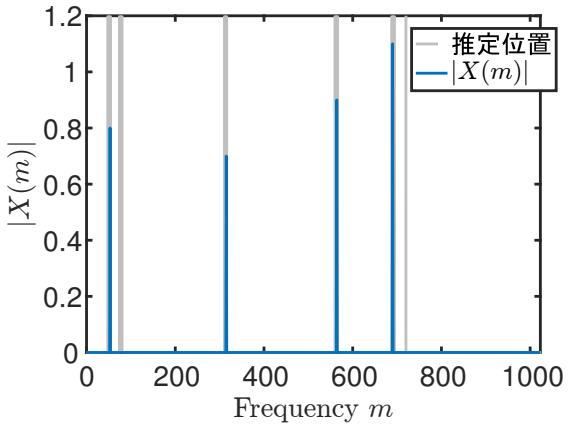

Fig. 8　より多く推定された位置

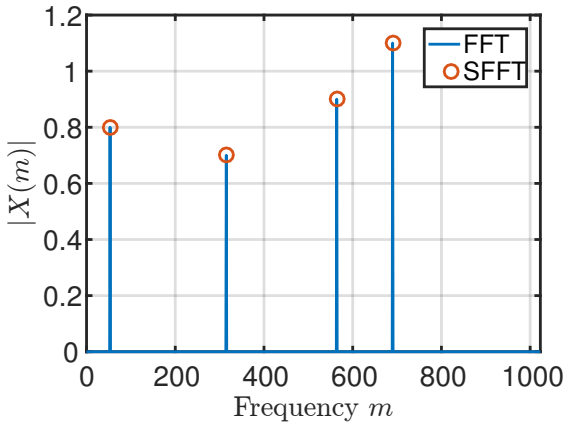

Fig. 9　入力と出力結果

スペクトルの位置を推定するためには，並び替えパラメータ r を変えて何度も位置の推定を行えばよい．十分な位置の推定精度を得るために必要な処理回数 L は経験的に 10〜20 回程度である．

入力信号の主要なスペクトルの位置を十分に絞り込んだ後に，フーリエ係数（スペクトルの大きさ）の推定を行う．周波数スペクトル $\boldsymbol{Z}$ と，窓関数の周波数スペクトル $\boldsymbol{W}$ を，推定した主要なスペクトルの位置によって対応させることでフーリエ係数の推定が可能である．また，精度を上げるために L 回の処理で得られた全ての $\boldsymbol{Z}$ に対して推定を行い，その中央値によって値を決定する．最終的な推定結果は Fig.9 のようになり，原信号の主要なスペクトルを正しく求められた．

5.　MATLAB による実装

本章では，実際に MATLAB で行ったスパースフーリエ変換の実装について説明する．次に示す Algorithm 1 ∼ 4 は，第 4 章のアルゴリズムをベースに，実際に MATLAB で実装したアルゴリズムを疑似コードを用いて表したものである．基本的な表記は MATLAB に従うが，一部数学的な表記に書き直している部分があるため注意が必要である．

Algorithm 1 の SFFT-PLAN 関数は，引数として入力信号 x，sparsity k，信号の SNR δ_{SNR} を与えると，他の関数を呼び出して全体を動かす中心的な関数である．

SETPARAM 関数は，信号長 N，sparsity k，信号の SNR δ_{SNR} に応じて，必要なパラメータを返す関数である．ここで，SFFT が最適な挙動をするように，予め返すパラメータを設定しておかなければならない．

MAKEFILTER 関数は，信号長 N，パラメータ B，窓関数のサポート S_{supp} に応じて，窓関数を設計する関数である．

SFFT 関数は，実際にスパースフーリエ変換を実行する関数である．13 行目 ∼20 行目までの処理は，予めスペクトルの位置を大まかに絞ることでその後の処理を高速化するための処理である [3)4)]．また，22 行目では L 個の並び替えパラメータ r によって，入力信号の並び替えと行列化を同時に行うことで，全体として L 回必要な位置推定を行列演算に置き換えている．これによって処理全体のパフォーマンスの向上を図っている．

Algorithm 1 Setup and Call SFFT

function SFFT-PLAN($x, k, \delta_{\mathrm{SNR}}$)
  $N = \mathrm{length}(x)$ ▷ N : 信号長
  $B = 2^{\mathrm{round}(\log_2(k))+3}$ ▷ B :Subsampled-FFT の点数
  $S_{\mathrm{supp}} = B\log_2(N) - 1$ ▷ S_{supp} : 窓関数のサポート長
  $[M, L, d, T_{\mathrm{thresh}}] =$SETPARAM$(N, k, \delta_{\mathrm{SNR}})$ ▷ パラメータ設定
  $[W_{\mathrm{time}}, W_{\mathrm{freq}}] =$MAKEFILTER$(N, B, S_{\mathrm{supp}})$ ▷ 窓関数設計
  $[I, A] =$SFFT$(x, B, M, L, d, T_{\mathrm{thresh}}, F_{\mathrm{time}}, F_{\mathrm{freq}}, S_{\mathrm{supp}})$ ▷ SFFT の実行
  return I, A
end function

Algorithm 2 Parameters for SFFT

function SETPARAM($N, k, \delta_{\mathrm{SNR}}$) ▷ パラメータは条件に応じて適切に設定
  $M = \frac{N}{2^2} \sim \frac{N}{2^{10}}(\mathrm{divides}\ N)$ ▷ M :Rough 推定で用いるパラメータ
  $L = 10 \sim 20$ ▷ L : 処理回数
  $d = k \sim 1.5k$ ▷ d :Subsampled-FFT の結果からスペクトルを選択する数
  $T_{\mathrm{thresh}} = 0.6L \sim 0.8L$ ▷ T_{thresh} : 結果から不要な位置を排除する際のしきい値
  return $M, L, d, T_{\mathrm{thresh}}$
end function

Algorithm 3 Make Filter for SFFT

function MAKEFILTER(N, B, S_{supp})
  $g = [\mathrm{chebwin}(S_{\mathrm{supp}}, 100)^{\mathrm{T}}, \mathrm{zeros}(1, N - S_{\mathrm{supp}})]$ ▷ g :Dolph-Chebyshev 窓
  $H = [\mathrm{ones}(1, \frac{N}{B}), \mathrm{zeros}(1, N - \frac{N}{B})]$ ▷ H : 矩形な周波数スペクトル
  $H = \mathrm{circshift}(H, [0, -\frac{N}{2B}])$
  $h = \mathrm{Re}(\mathrm{ifft}(W))$ ▷ h : H の時間信号
  $h = \mathrm{circshift}(h, [0, \mathrm{round}(\frac{S}{2} - 1)])$
  $W_{\mathrm{time}} = \mathrm{Re}(g\ .*\ h)$ ▷ W_{time} : 設計した窓関数の時間波形
  $W_{\mathrm{freq}} = \mathrm{fft}(W_{\mathrm{time}})$ ▷ W_{freq} : 設計した窓関数の周波数スペクトル
  return $W_{\mathrm{time}}, W_{\mathrm{freq}}$
end function

Algorithm 4 Sparse Fast Fourier Transform

function $\textsc{Sfft}(x, B, M, L, d, T_{\text{thresh}}, W_{\text{time}}, W_{\text{freq}}, S_{\text{supp}})$
  $N = \text{length}(x)$ ▷ 各変数の初期化
  $b = 1 : B$
  $m = 1 : \frac{N}{M} : N$
  $s = 0 : S_{\text{supp}} - 1$
  $y_{\text{p}} = \text{zeros}(L, B\text{ceil}(\frac{S_{\text{supp}}}{B}))$
  $z_{\text{p}} = \text{zeros}(L, B)$
  $J = \text{zeros}(1, \frac{dN}{M})$
  $J_{\text{p}} = \text{zeros}(1, \frac{d}{L})$
  $K = \text{zeros}(1, d)$
  $R = \text{zeros}(1, \frac{dN}{M})$
  $r = 2\text{floor}(\frac{N}{4}\text{rand}(L, 1)) + 1$ ▷ r : 並び替えに用いるパラメータ群
  $V = |\text{fft}(x(m))| + |\text{fft}(x(m+1))|$ ▷ ここから Rough 推定
  for $i = 1 : d$ **do** ▷ (詳細は文献 [3)4)] を参照)
    $[\tilde{}, K(i)] = \max(V)$
    $V(K(i)) = 0$
  end for
  for $i = 0 : \frac{N}{M} - 1$ **do**
    $R(di + 1 : d(i+1)) = K + Mi - 1$ ▷ R :Rough 推定結果
  end for ▷ ここまで Rough 推定
  $h = \text{mod}(\text{round}(\frac{B}{N} rR), B)$ ▷ h : 並び替え前後のインデックスを対応させる関数
  $y(:, s) = x(\text{mod}(rs, N)) .* \text{repmat}(W_{\text{time}}(s+1), L, 1)$ ▷ y : 並び替え · 窓かけ後の信号
  for $i = 0 : B : S_{\text{supp}}$ **do**
    $z(:, b) = z(:, b) + y(:, b + i)$ ▷ z : y の Sub 系列
  end for
  $Z = \text{fft}(z^{\text{T}})$ ▷ Z :Subsampled-FFT 結果
  $Z_{\text{abs}} = |Z|$
  for $i = 1 : d$ **do**
    $[\tilde{}, J_{\text{p}}(i, :)] = \max(Z_{\text{abs}})$ ▷ J_{p} : x_{p} に対する主要なスペクトルの推定位置
    $Z_{\text{abs}}(J_{\text{p}}(i, :) + (0 : B : BL - 1)) = 0$ ▷ 現在最大のスペクトルを消す
    for $l = 1 : L$ **do**
      $J = J + (h(l, :) == J_{\text{p}}(i, l) - 1)$ ▷ J : 主要なスペクトルの推定位置
    end for
  end for
  $I = R(J > T_{\text{thresh}})$ ▷ しきい値以上推定された位置のみ選択
  $A_{\text{est}} = \text{zeros}(L, \text{length}(I))$
  for $i = 1 : L$ **do**
    $o = \text{round}(\frac{B}{N} Ir(i))$ ▷ o : Z_{p} と F_{freq} のインデックスを対応させる関数
    $A_{\text{est}}(i, :) = Z_{\text{p}}(\text{mod}(o, B) + 1, i)^{\text{T}} ./ W_{\text{freq}}(\text{mod}(\frac{N}{B} o - r(i)I, N) + 1)$
  end for ▷ A_{est} : フーリエ係数の推定値
  $A = \text{median}(\text{Re}(A_{\text{est}}(:, :))) + \boldsymbol{j} \cdot \text{median}(\text{Im}(A_{\text{est}}(:, :)))$ ▷ A : フーリエ係数
  return I, A
end function

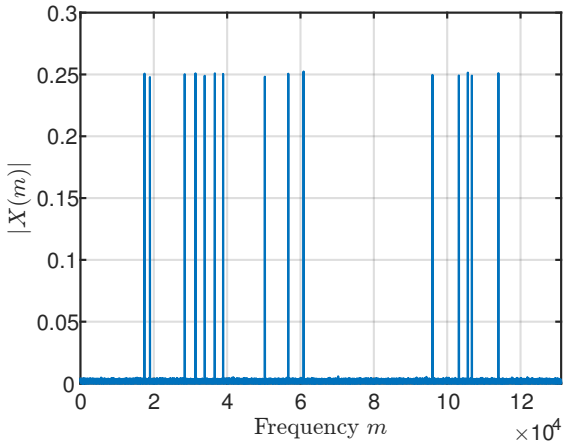

Fig. 10　実験用の信号の例 $(N = 2^{17}, k = 16)$

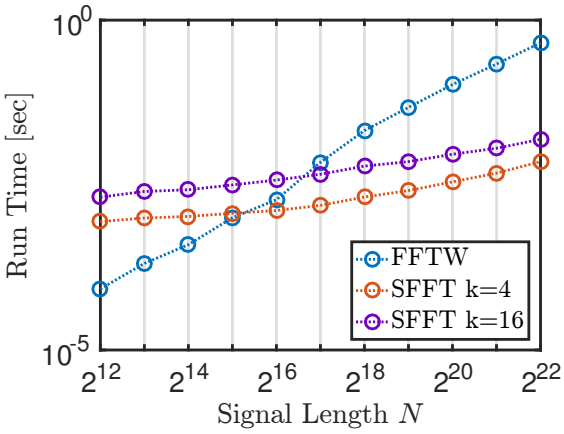

Fig. 11　処理時間の測定結果

6.　性能の評価

本章では，第 5 章で実装した MATLAB のプログラムを用いて実際にスパースフーリエ変換の性能評価を行い，その結果について考察する．6.1 節では，スパースフーリエ変換の速度評価を行う．6.2 節では，白色ガウス雑音を加えた複素スパース信号に対してスパースフーリエ変換を行い，主要なスペクトルの位置推定エラーの個数を測定する．ここで，位置推定エラーとは主要なスペクトルの誤推定を意味する．実験では，Fig.10 に示すような主要なスペクトルの強さがすべて等しいスパース信号を用いる．

6.1　速度評価

本実験では，複素スパース信号を用いて，スパースフーリエ変換の処理時間を測定する．実行速度の比較対象として，MATLAB 標準の FFT 関数を用いる．処理時間の測定は MATLAB の timeit 関数によって行う．速度計測に用いる実験環境を Table1 に示す．

実行速度の計測結果を図 11 に示す．スパースフーリエ変換は，信号が非常に長い場合に FFT より高速に実行可能であるという結果が得られた．一方で，スパースフーリエ変換の実行速度は sparsity k に依存し，k が小さいほどより高速に実行可能であることが確認できた．

6.2　位置推定エラー

本実験では，雑音を含む複素スパース信号を用いて，スパースフーリエ変換の位置推定の精度に関する実験を行う．信号に添加する白色雑音の分散を変化させ，位置推定エラーの個数を測定する．実験は各 SNR ごとに 10,000 回位置推定を行う．スパースフーリエ変換の内部のパラメータは，出力精度が良くなるように予め調整済みである．

位置推定エラーの測定結果を Fig.12 に示す．Number of Error は 10,000 回の推定で発生した位置推定エラーの個数で，Error Rate は位置推定エラーの発生率を表す．結果より，SNR = 0[dB] と雑音が非常に強い場合でも，位置推定エラーの発生率は 1%程度であり，十分な推定精度と，白色雑音に対するロバスト性があるという結果が得られた．

Table 1　速度計測で用いた実験環境

OS	Windows 10 Enterprise 64bit
CPU	Intel(R) Core 2 DUO CPU P8800 @ 2.66GHz x 2Core
Memory	4.00GB
Software	MATLAB R2016a 64-bit

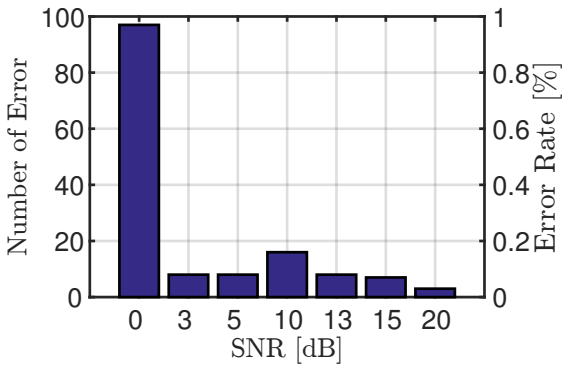

Fig. 12　位置推定エラーの測定結果

7. むすび

本論文では，スパース信号に対して高速にスペクトル推定が可能なスパースフーリエ変換のアルゴリズムについてまとめ，MATLAB において実装したプログラムを用いて位置推定精度と実行時間の評価を行った．スパースフーリエ変換の実行時間は sparsity に依存することが確認できた．信号が十分に長く sparsity が小さい場合，スパースフーリエ変換は高速フーリエ変換以上に高速に実行可能であることを示した．また，スペクトルの強さがすべて等しいスパース信号に対する推定精度は非常に良く，白色雑音に対して高いロバスト性があることを確認した．

参考文献

1) 樋口 龍雄，川又 政征: MATLAB 対応ディジタル信号処理，昭晃堂 (2000)

2) Dimitris G. Manolakis and Vinay K. Ingle: Applied Digital Signal Processing, Cambridge University Press (2011)

3) H. Hassanieh, P. Indyk, D. Katabi and E. Price: Simple and Practical Algorithm for Sparse Fourier Transform, Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms, 1183/1194 (2012)

4) Y. Mansour: Randomized interpolation and approximation of sparse polynomials stPreliminary version, Automata, Languages and Programing, **623**, 261/272 (2005)